Reti Neurali Artificiali per lo studio del mercato

Università degli studi di Brescia - Dipartimento di metodi quantitativi
Marco Sandri (sandri.marco@gmail.com)

Regressione non lineare con un modello neurale feedforward

Obiettivi:

(1) Imparare a stimare i pesi di una rete neurale feedforward con Matlab per affrontare un problema di regressione non lineare. Come si vede dalla spiegazione contenuta in queste pagine, con Matlab è piuttosto semplice. I comandi **`newff`**, **`train`** e **`sim`** sono i comandi fondamentali con cui, rispettivamente, si crea la rete, si stimano i suoi parametri con i dati e si effettuano poi previsioni. Attenzione però che, se da un lato Matlab semplifica di molto il codice necessario per stimare i pesi di una rete, rimangono sempre le difficoltà legate alle reti neurali:
- difficoltà connesse all'uso degli algoritmi di ottimizzazione numerica;
- difficoltà connesse alla costruzione della rete, in particolare alla scelta della sua complessità ottimale (che è legata al numero di neuroni dello strato nascosto); se questa complessità è insufficiente, la rete può non essere in grado di rappresentare adeguatamente il problema e quindi di fare buone previsioni; se invece è eccessiva, la rete rischia di cadere in overfitting, iperspecializzandosi sui dati, approssimando non solo la componente deterministica ma anche quella stocastica e quindi perdendo capacità di generalizzazione.

(2) Fare una serie di esperimenti numerici per comprendere il ruolo di uno dei parametri più importanti per una rete neurale: il numero di neuroni nello strato nascosto. Variando **`nhid`** potremo creare situazioni di under- e over-fitting, cioè situazioni in cui la complessità della rete è insufficiente o eccessiva per il problema che le si chiede di risolvere. Variando il numero di input **`ninp`** si potrà anche sperimentare il ruolo che hanno le variabili non importanti (dette talora *noisy variables*) sulla capacità previsiva della rete. Sarà interessante anche provare a lavorare con campioni piccoli o grandi (variando il parametro **`N`**) e con dati più o meno affetti da rumore (variando il parametro **`devstd`**).

Osservazioni:

(1) Nell'esempio che abbiamo qui considerato non analizziamo dei dati reali, ma dei dati simulati. Si tratta cioè di dati artificiali, da noi generati utilizzando un processo (non lineare) generatore dei dati piuttosto semplice:

$$y = -0.6 + 1.2 \cdot x_1^3 + \sin(3.46 \cdot x_1) + \varepsilon$$

dove ε è la componente stocastica distribuita come una normale di media nulla e deviazione standard s, cioè $\varepsilon \sim N(0, s)$. Si vede quindi che y è influenzata solo dalla variabile x_1. Tutte le altre variabili osservate (x_2, x_3, etc.) sono quindi variabili non importanti ai fini previsivi. Anzi, sono variabili che, se utilizzate in fase di stima, possono peggiorare la qualità della previsione del modello.

(2) Grazie al fatto che si tratta di dati simulati, noi conosciamo tutto di essi, anche la componente deterministica (indicata con una linea rossa), che nei problemi reali non è mai nota e che di fatto è il vero oggetto di indagine. Quindi in questo esperimento numerico abbiamo anche la possibilità di valutare se la rete riesce ad approssimare bene la componente deterministica. Nella realtà, lo ribadiamo, questa verifica ovviamente non è mai possibile e l'unica strada percorribile per valutare la bontà del modello è quella dell'analisi dei residui.

La scelta dell'architettura di una rete

La scelta dell'architettura della rete dipende prima di tutto dalla natura del problema che si intende risolvere.

- **Se si deve individuare un modello di regressione (non lineare), allora la scelta cade ad es. sulle reti feedforward;**
- **Se si vuole modellizzare/predire una serie temporale, allora si possono utilizzare le reti ricorrenti (ad es. le reti di Elman); in esse gli output ritardati sono input della rete;**
- **Se il problema invece è di stabilire a quale di un set di k classi le singole unità campionarie appartengono (classificazione) allora la scelta dovrà ad es. cadere sulle reti feedforward opportunamente adattate o sulle reti LVQ (learning vector quantization);**
- **Se il problema è di individuare nel campione, sulla base delle variabili osservate, dei gruppi omogenei di unità (cluster), allora si dovranno scegliere ad es. le SOM (self-organizing map).**

La scelta dell'architettura di una rete

I comandi riportati qui sulla destra creano un oggetto *neural network* ed impostano le sue proprietà in accordo con l'architettura da noi scelta.

Esiste però anche la possibilità di costruire architetture nuove non previste nel toolbox di Matlab e di effettuare poi la stima dei loro parametri (con il comando *'train'*) e di usarle per la previsione (con il comando *'sim'*).

New Networks Functions	
network	Create a custom neural network.
newc	Create a competitive layer.
newcf	Create a cascade-forward backpropagation network.
newelm	Create an Elman backpropagation network.
newff	Create a feed-forward backpropagation network.
newfftd	Create a feed-forward input-delay backprop network.
newgrnn	Design a generalized regression neural network.
newhop	Create a Hopfield recurrent network.
newlin	Create a linear layer.
newlind	Design a linear layer.
newlvq	Create a learning vector quantization network
newp	Create a perceptron.
newpnn	Design a probabilistic neural network.
newrb	Design a radial basis network.
newrbe	Design an exact radial basis network.
newsom	Create a self-organizing map.

La stima dei parametri di un modello neurale

Con il termine *'training'* si indica quello che in ambito statistico viene indicato con il termine di stima dei parametri della rete. La stima dei parametri di un modello equivale al problema della ricerca delle soluzioni di un problema di ottimo:

- trovare i parametri del modello che massimizzano la funzione di (log-)verosimiglianza;

- trovare i parametri del modello che minimizzano una funzione d'errore (mae, mse, msereg, etc..).

Indichiamo con $f(x, \theta)$ il nostro modello, dove x è il vettore di input (variabili osservate) e θ il vettore dei parametri del modello. La stima dei parametri consiste ad es. nella soluzione del problema di ottimo (minimi quadrati):

$$\min_{\theta \in \Theta} \sum_{i=1}^{N} (y_i - f(x_i, \theta))^2$$

La stima dei parametri di un modello neurale

A volte, specie con i modelli lineari, la soluzione al problema di ottimo la si può calcolare in modo semplice attraverso una formula. Ad esempio, nel modello di regressione lineare. la stima a minimi quadrati del vettore dei parametri è:

$$\beta^* = (X^T * X)^{-1} X^T Y$$

Nella maggior parte dei casi però la soluzione del problema di ottimo deve però essere trovata attraverso metodi numerici.
L'impiego di questi metodi richiede una buona conoscenza dei diversi algoritmi numerici sviluppati dai ricercatori e dei limiti insiti in ciascuno di essi.

Gli algoritmi numerici di ottimizzazione

La scelta dell'algoritmo di ottimizzazione si basa su diversi elementi:

- la natura e le dimensioni del problema di ottimo (no. di parametri da stimare, no. di dati nel campione,tipo di funzione da ottimizzare);

- la disponibilità di risorse computazionali (velocità del processore, memoria disponibile) e le corrispondenti richieste computazionali dell'algoritmo (velocità di convergenza, tempi di calcolo per ogni iterazione, quantità di memoria richiesta).

Gli algoritmi numerici di ottimizzazione

Gli algoritmi di ottimizzazione numerica sono in generale processi iterativi del tipo: $x_{t+1} = x_t + \Delta_t$

Δ_t esprime lo spostamento dell'algoritmo da x_t verso il punto di ottimo. Di questo spostamento di deve determinare direzione e ampiezza. Molti algoritmi calcolano Δ_t sulla base del gradiente della funzione e della sua Hessiana.

L'arresto dell'algoritmo di ottimizzazione avviene:

- dopo che e' trascorso un tempo massimo;

- dopo un numero massimo di iterazioni;

- quando l'errore e' sceso al di sotto di una soglia fissata;

- quando il gradiente e' sceso sotto una soglia fissata;

- quando l'errore sul set di validazione è aumentato più di un certo numero di volte dall'ultima volta che è diminuito (questo ovviamente quando si usa la cross-validation).

Gli algoritmi numerici di ottimizzazione

Nel "Neural Network Toolbox" di Matlab sono disponibili diversi algoritmi di ottimizzazione numerica, alcuni specificamente pensati per il training delle reti neurali (ad es. le numerose versioni di backpropagation: traingd, traingda, traingrp, etc).

Training Functions	
trainbfg	BFGS quasi-Newton backpropagation.
trainbr	Bayesian regularization.
traincgb	Powell-Beale conjugate gradient backpropagation.
traincgf	Fletcher-Powell conjugate gradient backpropagation.
traincgp	Polak-Ribiere conjugate gradient backpropagation.
traingd	Gradient descent backpropagation.
traingda	Gradient descent with adaptive lr backpropagation.
traingdm	Gradient descent with momentum backpropagation.
traingdx	Gradient descent with momentum and adaptive lr backprop.
trainlm	Levenberg-Marquardt backpropagation.
trainoss	One-step secant backpropagation.
trainrp	Resilient backpropagation (Rprop).
trainscg	Scaled conjugate gradient backpropagation.
trainb	Batch training with weight and bias learning rules.
trainc	Cyclical order incremental training with learning functions.
trainr	Random order incremental training with learning functions.

La stima di un modello neurale con Matlab

```
[Xn,psx] = mapminmax(X',-1,1);
[yn,psy]  = mapminmax(y',-1,1);
```

(1) Pre-processamento dei dati

```
net=newff(minmax(Xn),[nhid 1],{'tansig','tansig'},'trainlm');
```

(2) Creazione dell'oggetto *neural network*

```
net=init(net);
```

(3) Inizializzazione dei pesi della rete

```
optnet=train(net, Xn , yn);
```

(4) Stima dei pesi/parametri della rete

```
yout=sim(optnet, Xn);
```

(5) Previsione con la rete

```
yhat = mapminmax('reverse',yout,psy);
```

(6) Processamento inverso dei dati

Pre-processing dei dati

Talune trasformazioni dei dati di partenza possono essere utili per migliorare la successiva fase di stima dei parametri della rete. In Matlab sono disponibili 3 tipi di pre-processing dei dati:

(1) Riscalamento fra un min ed un max.

```
[Xn,psx] = mapminmax(X',-1,1);
```

Ciascuna riga di X viene riscalata fra -1 ed 1.

(2) Standardizzazione

```
[Xn,psx] = mapstd(X',0,1);
```

Ciascuna riga di X viene trasformata in modo da avere media 0 e deviazione standard 1.

(3) Analisi delle componenti principali

```
[Xn,psx] = processpca(X',0.1);
```

Processa la matrice X con la PCA in modo che le componenti scartate spieghino al più il 10% della variabilità totale dei dati